

Written Exam Correction (Preparation)

1 – Quizz

- 1) A superkey is
- the set of all attributes belonging to any candidate key
 - the set of attributes that has a unique value for each tuple in the relation
 - the primary key plus the secondary key
 - a minimal set of attributes that has a unique value for each tuple of the relation

***The only answer is b.
a and c make no sense.
d is the definition of a key.***

- 2) The allowed values for a foreign key are
- existing values from the referenced table
 - existing values from the referenced table or NULL
 - NULL
 - any value from the domain of the corresponding attribute

The only answer is a. NULL does not satisfy the foreign key constraint.

- 3) A candidate key is
- an attribute that is potentially the key of a relation
 - an attribute or a set of attributes that determines the values of all other attributes in the relation
 - an attribute with a unique value for each tuple of a relation
 - a minimal superkey
 - a set of attributes referring to another relation in the database

The only answer is d. A minimal superkey boils down to a candidate key.

- 4) If it is necessary to retrieve the values of an attribute for all tuples in a relation, which operator from the relational algebra should be used?
- equijoin
 - select
 - join
 - project

The only answer is d: project. Per definition.

- 5) What is the result of the project operation over attribute B applied on the given table R?

R

A	B	C
---	---	---

1	a	10
2	b	20
3	b	10
4	c	30
5	a	25
6	d	20

- a) A table with 3 attributes (A, B, C) and 4 tuples
 $\{ \langle 1, a, 10 \rangle, \langle 2, b, 20 \rangle, \langle 4, c, 30 \rangle, \langle 6, d, 20 \rangle \}$
- b) A table with 3 attributes (A, B, C) and 6 tuples:
 $\{ \langle 1, a, 10 \rangle, \langle 2, b, 20 \rangle, \langle 3, b, 10 \rangle, \langle 4, c, 30 \rangle, \langle 5, a, 25 \rangle, \langle 6, d, 20 \rangle \}$
- c) A table with 2 attributes (A, B) and 6 tuples
 $\{ \langle 1, a \rangle, \langle 2, b \rangle, \langle 3, b \rangle, \langle 4, c \rangle, \langle 5, a \rangle, \langle 6, d \rangle \}$
- d) A table with 2 attribute (A, B) and 4 tuples
 $\{ \langle 1, a \rangle, \langle 2, b \rangle, \langle 4, c \rangle, \langle 5, a \rangle, \langle 6, d \rangle \}$
- e) A table with one attribute (B) and 4 tuples
 $\{ \langle a \rangle, \langle b \rangle, \langle c \rangle, \langle d \rangle \}$
- f) A table with one attribute (B) and 6 tuples
 $\{ \langle a \rangle, \langle b \rangle, \langle b \rangle, \langle c \rangle, \langle a \rangle, \langle d \rangle \}$

The only result is e. Per definition.

- 6) Two tables are given: STUDENT(StudNo,Name,Department) and GRADES(StudNo,Course,Grade). What is the effect of the following relational algebra query: $\pi_{NAME}((\pi_{COURSE, STUDNO}(\text{GRADES}) \div \pi_{COURSE}(\text{GRADES})) \text{ join on StudNo STUDENT})$ where join on StudNo is the equijoin on the attribute StudNo.
- a) List students who have passed at least one course.
b) List students who have passed some courses.
c) List students who have passed no courses.
d) List students who have not passed all courses.
e) List students who have passed all courses.

e. The division finds the student number of the students that passed all courses. The equijoin associates the name with the student number.

- 7) Two tables are given: STUDENT(StudNo,Name,Department) and GRADES(StudNo,Course,Grade). Which of the given SQL statements is equivalent to the following relational algebra query:
 $\pi_{NAME}((\pi_{COURSE, STUDNO}(\text{GRADES}) \div \pi_{COURSE}(\text{GRADES})) * \text{STUDENT})$
- a) SELECT name
FROM student
WHERE NOT EXISTS
((SELECT course FROM grades)
EXCEPT
(SELECT course FROM grades
WHERE student.studno=grades.studno));

- b) SELECT name
FROM student
WHERE NOT EXISTS
(SELECT * FROM grades
WHERE student.studentno=grades.studno);
- c) SELECT name
FROM student
WHERE NOT EXISTS
((SELECT course FROM grades)
EXCEPT
(SELECT grade FROM grades
WHERE student.studno=grades.studno));
- d) SELECT name
FROM student
WHERE EXISTS
(SELECT course FROM grades
WHERE NOT EXISTS
(SELECT * FROM grades
WHERE student.studentno=grades.studno));
- e) SELECT name
FROM student
WHERE NOT EXISTS
(SELECT course FROM grades
WHERE NOT EXISTS (SELECT *
FROM grades
WHERE student.studentno=grades.studno));
- f) SELECT name
FROM student
WHERE EXISTS
(SELECT * FROM grades
WHERE student.studentno=grades.studno);

An answer is a. It selects those students for which the difference between all courses and the courses they have passed is empty.

b. Selects the students that have not passed a single class.

c. Is not well formed as except is expressed between two non union compatible relations.

d. Selects the students for which the difference between all courses and the courses they have passed is NOT empty.

Another answer is e. It selects the students for which the set of course they have not passed is empty

f. selects the students that have passed a course.

8) Is this a well formed XML document?

```
<?xml version="1.0"?>
<note>
<to age="29">Tove</to>
```

```
<from>Jani</from>
</note>
```

- a) Yes
- b) No

a. Yes this is a well formed document. 1 root (note). All the opened elements are closed in the correct order.

9) Is this a well formed XML document?

```
<?xml version="1.0"?>
<note>
<to age="29"/>
<from>Jani</from>
</note>
```

- a) Yes
- b) No

a. Yes this is a well formed document. One root, all the opened elements are closed. To is an empty element.

10) Consider the relation account(id int, balance int) and two transactions

T1: find the average balance over all accounts

T2: increase the balance by 5% for id equals 45

What statements are true:

- a) The order in which T1 and T2 does not impact the result of the average computed in T1
- b) T1 and T2 can be serialized
- c) T1 and T2 are conflicting

a. is not true. Whether the average is computed before or after the balance increase impacts the result.

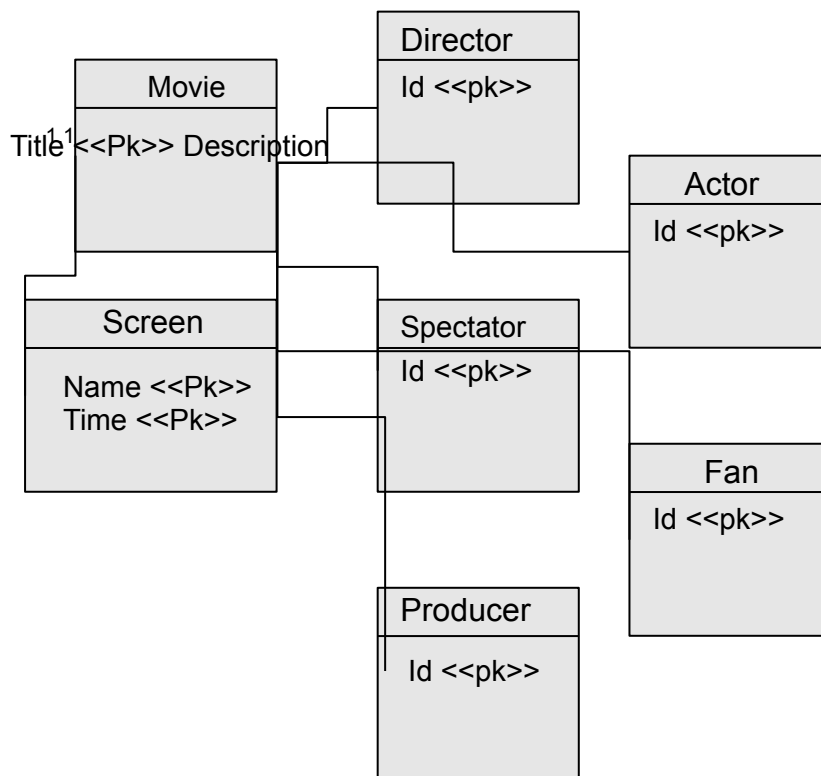
b. T1 and T2 can be serialized: T1,T2 or T2,T1 are serial execution.

c. T1 and T2 are conflicting: r/w operations on balance for id 45.

2 – Modelling, Relational Model, Query Languages

Question 1. UML Diagram

There are basically two approaches to this diagram. The first one is to map each relation into an entity class. The problem is that you don't get any associations. The second approach, drawn below, is to identify the entities which are referred to in the schema but are not materialized as relations, and to define associations that correspond to the relationship relations from the schema. All associations are n-n if no cardinality is explicitly given.



Question 2.

1. This is basically a foreign key constraint, where on deletion of a movie all references to it (via the key title) should be removed.
We should add to the create table statements for screen, see, play, produce and like the following constraint
FOREIGN KEY title references movie(title) on delete cascade
2. This is a constraint involving the relations movie and produce. We want to make sure that there does not exist a director that produced the movie she directed.

CREATE ASSERTION CHECK

NOT EXIST (SELECT title FROM movie m, produce p WHERE m.director = p.producer and m.title = p.title);

Question 3.

1. Find the actors that played in a movie directed by Fly
2. Not applicable
3. Find the producer that never produced a film that Philippe liked.

Question 4.

a) relational algebra and b) SQL

1. a. $\text{project}_{\text{actor}}(\text{thetajoin}_{\text{actor=produce}}(\text{play}, \text{produce}))$
b. $\text{select pl.actor from play pl, produce pr where pl.actor = pr.produce};$
2. a. like – see

- b. select * from like except select * from see;
3. a. project_{movie} (title) – project_{movie} (screen)
 b. select title from movie m
 where not exist (select s.title from screen s where s.title = m.title)
 NOTE: 3b and 2b are two equivalent ways of expressing a set difference in SQL.
4. Those producers that only see the movies they produce are all producers minus those who never see a movie and those that at least once see a movie made by another producer. Note that this reformulation allows us to rephrase a problem in terms of a subset of movie in terms of a problem concerning a single movie – and a straightforward formulation in both the relational algebra and SQL.
- a. producer_{nomovie} = project_{producer} (produce) – project_{spectator} (see)
 producer_{see} [s, title, p] = naturaljoin (produce, see)
 producer_{othermovie} = project_{producer} (join_{p != producer and s=producer} (producer_{see}, produce))
 project_{producer} (produce) – producer_{nomovie} – producer_{othermovie}
- b. select p.producer from produce p
 where p.producer not in (select pp.producer from produce pp
 where pp.producer not in (select spectator from see))
 and p.producer not in (select pp.producer from produce pp, produce ppp, see s
 where pp.producer != ppp.producer
 and ppp.title = s.title
 and pp.producer = s.spectator);

Question 5.

1.

```
<?xml version="1.0"?>
<movies>
<movie title="Schrek" >
<description>animation</description>
<director>Chris Miller</director>
</movie>
<movie title="The incredibles" >
<description>animation</description>
<director>Brad Bird</director>
</movie>
</movies>
```

2.

```
//movie[./description = "animation"]/@title
```